Summary: This paper evaluates classification speed and performance on the MNIST dataset and provides recommendations to weigh the speed and performance of 2 alternate approaches.

Research design: The **first** approach (1a) fits 784 features of 10 handwritten digits to the first 60,000 (60k) of 70k labelled rows to train a Random Forest classifier (**RFC**), (1b) which is tested on the remaining 10k rows. The **second** approach (2a) applies Principal Component Analysis (**PCA**) to reduce the feature count while retaining 95% of the explained variation on the full 70k rows. (2b) The first 60k of these reduced rows then train a second **RFC** model, which is (2c) tested on the final 10k rows of the reduced set. Wall clock times are collected for steps (1a), (2a), and (2b). The tests of these 2 approaches, in steps (1b) and (2c), are evaluated for performance via a harmonic mean equally weighting precision and recall but weighting this F1 average performance by each set's 10 labels' proportions. The 1st two rows and 4 columns of **table 1** present these speed / performance results. A third approach corrects the second experiment's flawed train / test regimen by (3a) limiting the PCA transform method access to the first 60k rows only, and then (3b) fitting those rows to a third RFC. The 10k untouched test rows are then (3c) projected via the PCA onto the hyper plane of principal components derived in (3a) earlier, and (3d) that reduced dimension set of 10k rows tests the performance of the third RFC model fit, again via F1, presented with speed in **table 1** row 3.

Table 1: times and F1 performances for 3 experiments

	pca	rfc	tot time	f1 wtd avg	0	1	2	3	4	5	6	7	8	9
rfc		3.56768	3.56768	0.949160	0.967000	0.983736	0.940334	0.926543	0.946835	0.940774	0.964323	0.953695	0.930160	0.933602
pca70+rfc60	3.21958	7.15179	10.3714	0.896022	0.931440	0.977273	0.882892	0.865643	0.885017	0.853728	0.923077	0.907588	0.842849	0.876025
pca60+rfc60	2.45598	7.16778	9.62376	0.900539	0.933399	0.978556	0.888995	0.875842	0.885872	0.860904	0.927187	0.915438	0.857903	0.867384
data				100.000000	9.861429	11.252857	9.985714	10.201429	9.748571	9.018571	9.822857	10.418571	9.750000	9.940000
train				85.714286	9.871667	11.236667	9.930000	10.218333	9.736667	9.035000	9.863333	10.441667	9.751667	9.915000
test				14.285714	9.800000	11.350000	10.320000	10.100000	9.820000	8.920000	9.580000	10.280000	9.740000	10.090000

Python code: A 2018 MacBookPro with 6-core, single 2.9 ghz processor, 32 gb 2400 mhz DDR4 runs Python v3.68 in Jupyter notebook. Methods are principally Scikit-Learn (sklearn) with numpy and pandas assisting. MNIST version 1 data are loaded and assigned to X for explanatory and y label variables, and then to their train and test subsets employing the first 60,000 rows and next 10,000 rows. Train and test each are complete sets containing similar but not same proportions of every label, as shown in the final 2 rows x final 10 columns of **table 1**. All three experiments initialize RFC **max_features**='sqrt', **n_estimators**=10, **bootstra**p=True (default), initialize PCA **n_components**=0.95 and for control initialize RFC and PCA **random_state**=42. Time library's **time method** measures start and stop time. (1a) The first RFC model's **fit method** is applied to the train set. (1b) This RFC model's **predict method** computes predicted labels from the untouched test set's full 784 features and those values are compared with the test set's labels for performance. From sklearn's **classification report**, each experiment retains the F1 weighted average across digits and per digit, found in the first-row x final 11 columns of **table 1**. (2) In the 2nd experiment, (2a) a first PCA instance's **fit_transform method** fits 784 principal components and transforms 70k rows reducing explained variance to 95% and feature count from 784 to 154. (2b) The first 60k of these 70k reduced-feature rows are trained via a second RFC instance's **fit method**. (2c) The same RFC's **predict method** is applied to the final 10k reduced rows and compared with test labels. Time measures PCA and RFC separately and the F1 scores are retained. (3) Correcting the 2nd experiment's flaw, in (3a), only the train set's 60k rows are shown a 2nd PCA **fit_transform method** and then in (3b) to a 3rd RFC **fit method**. In (3c) the second PCA instance's **transform method** projects the 10k untouched test rows onto the 154-dimension hyper plane generated in (3a). (3d) The 3rd RFC **predict method** employs

these (reduced) test rows to predict labels that are compared with actual labels. Finally, experiment 3 is collected into a defined function 'pca_rfc_sense' in order to collect tuples of (F1 measure, time elapsed) for 18 explained variance levels between 5% and 95%.

Comparative results: Speed and performance are summarized in **table 1's** first 3 columns and final 11 columns, respectively. Total time for experiment 1 is about 3x as quick as experiments 2 and 3 for 2 reasons: (1) PCA contributes ~3.2 seconds for 70k rows and ~2.4 seconds for 60k rows in experiments 2 and 3 respectively. (2) RFC time doubles from 3.6 to 7.2 seconds when PCA is employed in experiments 2 and 3. F1 performance is best for experiment 1 deploying RFC only (~0.95) and worse when RFC is applied to PCA reduced sets (~0.90), slightly better for the 3rd corrected than 2nd flawed experiment. Experiment 1's outperformance is not surprising given the RFC in the 1st experiment had 5 times as many features to choose from. The performance impact of the flaw in the 2nd experiment can be examined by comparing **table 1's** rows 2 and 3 showing performance scores by digit and rows 4, 5 and 6 showing the proportion of digits in train and test set. Experiment 2 allows the digit proportions composition of the test to leak into the PCA transformation of the train set that is then fed into the RFC. One might expect the 2nd RFC to outperform the 3rd RFC considering more data in experiment 2 trained the PCA than in experiment 3, and that additional data ultimately was tested in step (3c). It would be difficult to untangle whether PCA general underperformance or specific aspects of the test set digit composition contribute to the slight underperformance of experiment 2 versus 3. For example, the digit 2 (7th column of **table 1** makes up 10% of the whole and train sets (4th and 5th rows) and 10.3% of the test set (6th row) but both the 2nd and 3rd experiments performed similarly on this digit. Digit 3 in the 8th row has a slightly smaller proportion in the test than

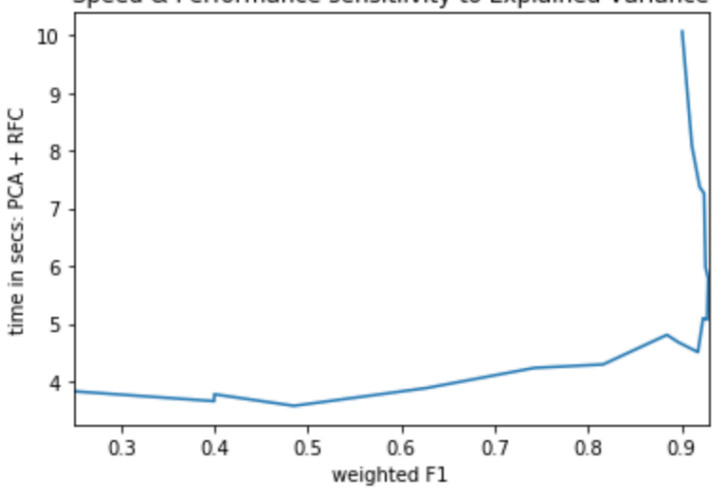
train set (10.1% versus 10.2%) which when included in experiment 2 caused slightly worse performance for this digit. Perhaps RFC performs relatively poorly when fitted to a reduced digit 3 versus fitting other digits on reduced sets.

Management recommendations. Shorter times and higher F1 are preferred. Thus, when employing RFC to classify MNIST digits, PCA dimension reduction in this configuration is not recommended for identically configured computer vision models. Whether PCA reduction would benefit speed and performance measures for other configurations depends upon datasets, classification methods, and performance measures including differently balanced F measures. The following **table 2** and **figure 1** demonstrate varying PCA explained variance may improve speed and performance, though not better than that of the RFC model alone: 70% explained variance obtains 92.8% F1 in just 5.8 seconds and 50% explained variance obtains 90% F1 in just 4.5 secs which approaches the RFC model 3.6 sec speed in experiment 1.

Table 2:
F1 and time at Explained Variances

	F1	secs
0.05	0.249071	3.825960
0.10	0.399521	3.654933
0.15	0.399521	3.774609
0.20	0.484875	3.574024
0.25	0.626302	3.880951
0.30	0.742041	4.231121
0.35	0.815815	4.291736
0.45	0.884266	4.804868
0.50	0.896474	4.676723
0.55	0.917347	4.505103
0.60	0.922969	5.095792
0.65	0.927293	5.067381
0.70	0.928364	5.782687
0.75	0.925484	5.981158
0.80	0.923995	7.258676
0.85	0.919320	7.365783
0.90	0.910917	8.087716
0.95	0.900539	10.068991

Figure 1:
Speed & Performance sensitiivity to Explained Variance



PCA may benefit comprehension of an underlying model with many features. 784 features may cluster together on their own, but be more comprehensible were their numbers reduced.